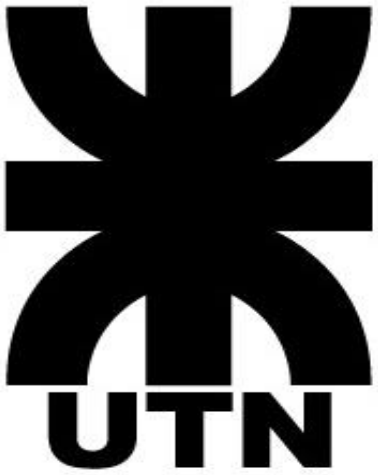

Universidad Tecnológica Nacional
Facultad Regional Concepción del Uruguay

Ingeniería en Sistemas de Información

Sistemas Operativos

Trabajo Práctico Final:
"Ampliación y mejora del shell experimental"

Integrantes del Grupo:

Arrúa, Martin
Lazbal, David
Rivera, Ramiro

Fecha de Entrega:
19/02/2015

Docentes:

Ing. Arellano Gabriel
Ing. Aguiar Osvaldo

Consigna.

Denominación

"Ampliación y mejora del shell experimental"

Características

Formando grupos de como máximo 3 integrantes y basándose de alguno de los shells desarrollados en el ciclo lectivo 2013:

- https://code.google.com/p/gst-shell/
- https://code.google.com/p/pascal-shell/
- https://code.google.com/p/shell-experimental-oshell/

Deberán analizar/mejorar el código del proyecto seleccionado y asegurarse que el mismo:

- Muestre un prompt y reciba las solicitudes del usuario e informe las salidas (normales y de error) de los programas.
- Permita la ejecución de programas (en el directorio actual o vía una ruta absoluta o relativa).
- Incluya las siguientes funcionalidades a través de comandos internos: pwd, cd, cat (concatenar hasta dos archivos, o si se pasa sólo uno, concatenar con la salida estándar), ls (incluyendo las opciones -l, -f, -a), kill.
- Operador pipe y de redirección de la salida estándar.
- Ejecución en segundo plano (incluyendo algo similar a los comandos fg y bg).

Introducción.

Este proyecto tiene como objetivo mejorar una de las shells experimentales realizadas durante el anterior ciclo lectivo, como parte de la cátedra Sistemas Operativos.
Nuestra primer tarea fue analizar cada uno de los trabajos y ponernos de acuerdo en cuál sería el elegido para realizar las mejoras e incluir nuevo funcionamiento. Estos trabajos tenían como objetivo crear una interfaz de usuario capaz de realizar algunas de las instrucciones básicas que la Shell nativa de Linux lleva adelante, como por ejemplo "cd", para navegar entre directorios, "pwd", mostrando el directorio actual, o "ls", que nos muestra el contenido del directorio en el que estamos al momento de ejecutarlo.
Luego del estudio, nuestra elección fue trabajar con la shell denominada OShell, desarrollada por Fernando Gómez Albornoz. Una vez detectados los errores, problemas y cosas a mejorar, nos pusimos a trabajar.

Principales mejoras.

A continuación se enunciaran los cambios más destacables del proyecto, con respecto a su versión anterior:

1. Se realizaron importantes mejoras en el comando "ls", principalmente en cuanto a su código, dado que el mismo era muy reiterativo, ya que dicho comando cuenta con la posibilidad de usar distintos argumentos que alteran el resultado final. Por ejemplo, "ls -a" muestra el contenido del directorio, pero ordenando el mismo por nombre, e indicando el tipo de archivo listado. Se redujo la cantidad de procedimientos
2. Se mejoró la opción "-l" del comando "ls". Como resultado, el mismo muestra una lista del contenido del directorio, pero con información más completa y mayores detalles.
3. Se eliminaron los distintos usos de la unit "crt", en pos de una mejor implementación, ya que la misma generaba problemas, por ejemplo, en el uso de la redirección, uno de los nuevos ítems con los que cuenta el proyecto.
4. Redirección de salida estándar: esta función nos brinda la posibilidad de guardar en un archivo específico la salida de otro comando, por ejemplo un "ps". Permite tanto la salida "destructiva" (>) como la "no destructiva" (>>).
5. Uso del operador "&": permite iniciar un nuevo proceso en segundo plano.
6. Implementación de control de trabajos:
   - Comando Jobs: el mismo se utiliza para listar los procesos iniciados mediante la terminal de usuario, que se estén ejecutando en segundo o primer plano.
   - Comando bg: permite la reanudación de la ejecución en segundo plano de un proceso previamente detenido mediante ctrl + z. Implementación: bg %[job_number].
   - Comando fg: permite la reanudación de la ejecución en primer plano de un proceso previamente detenido mediante ctrl + z. Implementación: fg %[job_nomber].

Página MAN.

```
ALROShell(1)                ALROShell                              ALROShell(1)


NAME
       ALROShell



SYNOPSIS
       ALROShell



COPYRIGHT
        ALROShell is Copyright (C) 2013-2015 Free Software Foundation, Inc.



DESCRIPTION
       ALROShell es una shell experimental que lleva adelante algunas de las
principales fun-
       ciones de la shell nativa de Linux.
       ALROShell fue modificada a partir de OShell, como parte del Trabajo Prác-
tico Final  de
       la  cátedra  Sistemas  Operativos  de la carrera Ingeniera en Sistemas de
Información,
       Universidad Tecnológica Nacional, Facultad Regional Concepción del Uru-
guay.



SHELL BUILTIN COMMANDS
       ls [ruta]
              Lista en pantalla los archivos del directorio actual o
              del directorio "ruta". Las opciones -a, -f y -l se
              pueden combinar.

       ls -a [ruta]
              Lista en pantalla los archivos del directorio actual o
              del directorio "ruta", incluyendo los archivos
              ocultos, ordenandolos alfabéticamente e indicando el
              tipo de archivo. Las opciones -a, -f y -l se pueden
              combinar.


       ls -f [ruta]
              Lista en pantalla los archivos del directorio actual o
              del directorio "ruta", sin indicar de que tipo son
              y sin ordenarlos alfabeticamente. Las opciones -a, -f
              y -l se pueden combinar.
```

```
ls -l [ruta]
        Lista en pantalla los archivos del directorio actual o
        del directorio "ruta", en formato de lista, indicando
        gran cantidad de información de cada archivo. Las opciones
        -a, -f y -l se pueden combinar.

pwd
        Muestra en pantalla la ruta del directorio en el que estamos.
        posicionados.

cd [ruta]
        Cambia el directorio de trabajo actual al directorio "ruta".


kill -id_señal pid
        Envia la señal "id_señal" al proceso "pid". Ambos parametros
        deben ser de formato numerico.

cat archivo1 [archvo2]
        Concatena "archivo1" con "archivo2".
        Si solo se le pasa un archivo, lo concatena
        con la salida estandar.


bg num_proc
        Envía el proceso "num_proc" a segundo plano.


fg num_proc
        Trae el proceso "num_proc" a primer plano.


jobs
        Muestra una tabla de los procesos iniciados
        mediante la terminal,en primer o segundo plano,
        indicando por ejemplo su estado actual.


exit
        Cierra la terminal.


moo [ -h | texto ]
        Los usuarios de Linux saben a que nos referimos.
        Prueben con 'moo -h' en caso contrario ;)



REDIRECTION
    Redireccion de Salida estandar


    comando > archivo
        Guarda la salida de "comando" en el "archivo". Si el
        archivo no existe, lo crea, y si existe lo reescribe.


    comando >> archivo
        Guarda la salida de "comando" en el "archivo". Si el
```

```
               archivo no existe, lo crea, y si existe lo agrega al final.


OPERATORS
       &
               Su uso al final de una instruccion iniciará
               la misma en segundo plano


PIPE
       comando1 | comando2

               El comando pipe envia la salida del primer como entrada del se-
gundo comando.
```


AUTHORS
       Ramiro Rivera - ramarivera@gmail.com
       David Lazbal - davidlazbal@gmail.com
       Martín Arrúa - martin94.profugo@gmail.com
       Fernando Gómez Albornoz - fgalbornoz07@gmail.com

Código.

```
{
      ALROShell.pp

      Copyright 2015
      * Ramiro Rivera                          <ramarivera@gmail.com>,
      * David lazbal                           <davidlazbal@gmail.com>,
      * Matín Arrúa                            <martin94.profugo@gmail.com>.

      Copyright 2014
      * Fernando Gómez Albornoz          <fgalbornoz07@gmail.com>.


      This program is free software; you can redistribute it and/or modify
      it under the terms of the GNU General Public License as published by
      the Free Software Foundation; either version 3 of the License, or
      (at your option) any later version.

      This program is distributed in the hope that it will be useful,
      but WITHOUT ANY WARRANTY; without even the implied warranty of
      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
      GNU General Public License for more details.

      You should have received a copy of the GNU General Public License
      along with this program; if not, write to the Free Software
      Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston,
      MA 02110-1301, USA.
}

Program ALROShell;
Uses BaseUnix, sysutils, unix, comandos, analizador, utilidades,ALR;

var salir: boolean;
      ENTRADA: string;

Begin
pidEnEjec:=-1;                          //Inicializacion
instalarManejadores;                //Instalacion de manejadores de seniales
crearTabla;
writeln('Bienvenido a ALROShell          -Copyright 2013-2015-');
salir:= false;                          // Break del ciclo repeat.

      repeat
            alBG:=false;
            write(CR,copy(TimeToStr(Time),1,5),' ',fpGetEnv('USER'),'# ');    //
Fecha y usuario (prompt)
            if length(strSin(dirActual,dirHome)) <= 20 then
                  Begin
                        if copy(dirActual,1,length(dirHome)) = dirHome then
write('~',strSin(dirActual,dirHome),' ') // Directorio (prompt)
                        else write('~',dirActual,' ');
                                                                        // Di-
rectorio (prompt)
                  end
            else  write('~/...',rightStr(strSin(dirActual,dirHome),20),' ');
                                          // Directorio (prompt)
```

```
            readln(ENTRADA);
            limpiarTabla;
            if not(analizar(ENTRADA)) then salir:= true;
      until salir = true;
      writeln();
      writeln('Esta shell no tiene Poderes de Super Vaca');
      writeln('(O si los tiene?)');
      writeln();
End.


UNIT analizador;
INTERFACE
      Uses  errors,ALR, comandos, utilidades, unix, baseunix, sysutils;

function analizar       (var ENTRADA: string): boolean;          // Analiza la
cadena introducida por el usuario.
procedure analizarCAT   (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando CAT.
function analizarCD           (DIRECTORIO: string): cint;              // Ana-
liza la cadena para ejecutar el comando CD.
function analizarEXEC   (var ENTRADA:string): cint;              // Analiza la
cadena para ejecutar el comando EXEC.
procedure analizarKILL  (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando KILL.
procedure analizarLS    (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando LS.
procedure analizarPWD   (var ENTRADA: string);                   // Analiza la
cadena para ejecutar el comando PWD.
procedure analizarReDIR (var ENTRADA: string);                   // Analiza la
cadena si presenta redirección de la salida estándar.
procedure analizarPipe  (var ENTRADA: string);                   // Analiza la
cadena si presenta pipes (tuberías).
procedure analizarBG    (var ENTRADA: string);                   // Analiza la
cadena si presenta solicitud de ejecutarse en 2do plano.


IMPLEMENTATION

function analizar(var ENTRADA: string): boolean;
var COMANDO, ARGUMENTOS:string;
Begin
      analizar := true;
      COMANDO     := extraerComando(ENTRADA);              //se extrae el co-
mando para el case
      ARGUMENTOS:= ENTRADA;                                //se obtienen los ar-
gumentos del comando
      if ARGUMENTOS <> '' then           //se vuelve a la entrada original
            ENTRADA := COMANDO + ' ' + ARGUMENTOS
      ELSE
            ENTRADA := COMANDO;
      COMANDO     :=upcase(COMANDO);

      if COMANDO='EXIT' then analizar:= false        // Salir del programa.
      else
            begin
            if COMANDO='JOBS' then mostrarTabla            // Muestra las
tareas.
            else
```

```
                begin
                if COMANDO='HELP' then help                // Muestra una panta-
lla de ayuda.
                else
                      begin
                      if solicitudBG(COMANDO) then bg(ARGUMENTOS)
      // Solicita ejecutar un proceso en segundo plano.
                      else
                            begin
                            if solicitudFG(COMANDO) then fg(ARGUMENTOS)
           // Solicita ejecujar un proceso en primer plano.
                            else begin
                                  if enBg(ENTRADA) then begin analizarBG(EN-
TRADA); writeln('esto dice que puse en &') end       // Envía al segundo plano.
                                  else begin
                                        if    redireccion(ENTRADA) then anali-
zarReDIR(ENTRADA)       // Redirección de la salida estándar.
                                        else begin
                                              if tuberia(ENTRADA) then anali-
zarPipe(ENTRADA)              // Tubería (pipe).
                                              else begin
                                                    if COMANDO='CAT' then ana-
lizarCAT(ARGUMENTOS)          // Comando CAT
                                                    else begin
                                                          if COMANDO='CD' then
analizarCD(ARGUMENTOS)  // Comando CD
                                                          else begin
                                                                if CO-
MANDO='KILL' then analizarKILL(ARGUMENTOS)      // Comando KILL
                                                                else begin
                                                                      if CO-
MANDO='LS' then analizarLS(ARGUMENTOS)    // Comando LS
                                                                      else
begin
                                                                            if
COMANDO='PWD' then analizarPWD(ENTRADA)   //Comando PWD
      else
      begin
      if COMANDO='MOO' then moo(ARGUMENTOS)
      else
            begin
            if ENTRADA <> '' then         // Comando externo
                  Begin
                  if analizarEXEC(ENTRADA) = -1 then
                        begin
                        mostrarerror(5);
                        exit;
```

```
                end;                    //Comando no reconocido.

            end;

        end;

        end;
                                                                    end;
                                                                end;
                                                            end;
                                                        end;
                                                    end;
                                                end;
                                            end;
                                        end;
                                    end;
                                end;
                            end;
                        end;
                    end;
                End;
end;


procedure analizarCAT(var ENTRADA: string);
var dir1, dir2: string;
        tipo: byte;
Begin
        if strOutPut = '' then tipo := 0 else tipo := 1; // Tipo 0: Salida están-
dar, tipo 1: Redirigir la salida.
        dir1 := extraerComando(ENTRADA);                    // se hace reuso de
esta funcion debido al analisis sintáctico, se obtiene el directorio del primer
archivo
        if dir1 = '' then
                begin
                mostrarerror(1);
                exit;
                end
        else
                begin
                        dir2 := ENTRADA;                            // se asigna a
dir2 la direccion del segundo archivo
                        if dir1[1] <> '/' then dir1 := dirActual+'/'+dir1;
                        if (dir2 <> '') and (dir2[1] <> '/') then dir2 := dirAc-
tual+'/'+dir2;
                        CAT (dir1,dir2,tipo);
                end;
End;



function analizarCD(DIRECTORIO: string): cint;
{
        Nombre: analizarCD.
        return: -1 si hay error, 0 si se ejecutó con éxito.
}
Begin
        analizarCD:= 0;
        if (DIRECTORIO='') then
                fpChDir(dirHome)
```

```
      else
            begin
                  if fpChDir(DIRECTORIO) <> 0 then
                              analizarCD:= -1;
            end;
End;



function analizarEXEC(var ENTRADA:string): cint;
{
   Nombre: analizarEXEC.
   Condición: No se permiten más de 3 parámetros.

}

var comando : string;
      aux         : parametros;
      tipo  : byte;
      i           : integer;
Begin
      analizarExec := 0;
      if (strOutPut = '') then tipo:= 0 else tipo:= 1;
            comando:=extraerComando(ENTRADA);         //ENTRADA vuelve sin el co-
mando.
            aux:=extraerArgumentos(ENTRADA,i);        //AUX se le asignan los pa-
rámetros de forma -* y ENTRADA contiene cualquier parametro sin la forma -*,
como podría ser un directorio
            if not(fileExists(comando)) then
                  comando:=FSearch(comando,strpas(fpGetenv('PATH')));
            if comando <> '' then
                  begin
                        if (ENTRADA = '') then        // Sólo existen parámetros
de la forma -*.
                              begin
                                    case i of   // 0, 1, 2 o 3 parámetros.
                                          0: EXEC(comando,[],tipo);
                                          1: EXEC(comando,[aux[1]], tipo);
                                          2: EXEC(comando,[aux[1],aux[2]],
tipo);
                                          3: EXEC(co-
mando,[aux[1],aux[2],aux[3]], tipo);
                                    end;
                              end
                        else                          // Existe algún parámetro
sin la forma -*, como podría ser un directorio
                              begin
                                    case i of   // 0, 1, 2 o 3 parámetros + algún
directorio o parámetro extra..
                                          0: EXEC(comando,[ENTRADA], tipo);
                                          1: EXEC(comando,[aux[1],ENTRADA],
tipo);
                                          2: EXEC(comando,[aux[1],aux[2],EN-
TRADA], tipo);
                                          3: EXEC(co-
mando,[aux[1],aux[2],aux[3],ENTRADA], tipo);
                                    end;
                              end
                  end
```

```
            else
                  analizarExec:= -1;
End;



procedure analizarKILL(var ENTRADA: string);
{
   Nombre: analizarKILL.
   Condición: Deben pasarse dos parámetros, sin excepción. Forma: kill -n_signal
pid

}

var argumentos: parametros;
      proc,signal: longint;
      i: integer;
      err: word;
Begin
      argumentos:= extraerArgumentos(ENTRADA,i);                  //se separan la
señal y el pid recibidos (pid en entrada)
      if i <> 1 then
      //verifica que se pase sólo 1 señal
            begin
            mostrarerror(8);  exit;
            end
      else
      begin
            val(argumentos[1],signal,err);
      //se asigna a signal el numero de señal recibido
            if err<>0 then
                  begin
                  mostrarerror(9);  exit;
                  end
            else
                  begin
                        val(ENTRADA,proc,err);
      //se asigna a signal el numero de señal recibido
                        if err<>0 then
                              begin
                              mostrarerror(9);
                              exit;
                              end
                        else
                              KILL(signal,proc);
                  end;
      end;
End;

procedure analizarLS(var ENTRADA: string);
{
   Nombre: analizarLS.
   Opciones: Puede haber 0, 1, 2 o 3 parámetros identificados con un guión <->
                  y pueden estar en cualquier orden. Estos son <-a>, <-f>, <-
l>.
                  Puede haber o no 1 argumento que indique la ruta desde la
cual
                  trabajar, luego de los parámetros comenzados con guión.
```

```
}

var
      directorio,cad,dirBase  : string;
      i,j                              : integer;
      argumentos                  : parametros;

Begin
      dirBase:= dirActual;
      argumentos:= extraerArgumentos(ENTRADA,i);
      for j:=1 to i do
            argumentos[j]:=upcase(argumentos[j]);
      directorio:= ENTRADA;
      case i of
      0:          begin
                  if (directorio <> '') then
                        cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                  else
                        cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                  if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                        ls(false,false,false)               //sí = 0 se posicionó
bien y ejecuta ls
                  else
                        begin
                        mostrarerror(4);  exit;
                        end                                       //si <> 0 el
directorio recibido no es coherente.
                  end;

      1:          begin
                  if not(paramValido(argumentos[1])) then
                        begin
                              mostrarerror(8); exit;
                        end;

                  if (directorio <> '') then
                        cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                  else
                        cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                  if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                        ls(                                             //si = 0
se posicionó bien y ejecuta ls
                              pos('A',argumentos[1])<>0,    //verifica si el ar-
gumento contiene el parametro 'A'
                              pos('F',argumentos[1])<>0,    //verifica si el ar-
gumento contiene el parametro 'F'
                              pos('L',argumentos[1])<>0)    //verifica si el ar-
gumento contiene el parametro 'L'
                  else
                        begin
                        mostrarerror(4);  exit;
                        end                                       //si <> 0 el
directorio recibido no es coherente.
```

```
                end;

      2:        begin
                if not(paramValido(argumentos[1])) or not(paramValido(argumen-
tos[2])) then
                      begin
                            mostrarerror(8); exit;
                      end;

                if (directorio <> '') then
                      cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                else
                      cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                      ls(                                                 //sí = 0
se posicionó bien y ejecuta ls
                            ((pos('A',argumentos[1])<>0) or (pos('A',argumen-
tos[2])<>0)),     //verifica si alguno de los argumentos contiene el parametro
'A'
                            ((pos('F',argumentos[1])<>0) or (pos('F',argumen-
tos[2])<>0)),     //verifica si alguno de los argumentos contiene el parametro
'F'
                            ((pos('L',argumentos[1])<>0) or (pos('L',argumen-
tos[2])<>0)))     //verifica si alguno de los argumentos contiene el parametro
'L'
                else begin
                      mostrarerror(4);
                      exit;
                      end;                          //si <> 0 el directorio re-
cibido no es coherente.
                end;

      3:        begin
                if not(paramValido(argumentos[1])) or not(paramValido(argumen-
tos[2])) or not(paramValido(argumentos[3])) then
                      begin
                            mostrarerror(8); exit;
                      end;

                if (directorio <> '') then
                      cad:= (directorio)            //si existe el directorio
se lo prepara para cd
                else
                      cad:= (dirBase);        //si no existe se usará el direc-
torio actual
                if analizarCD(cad) = 0 then               //se posiciona en el
directorio y verifica la salida
                      ls(                                                 //si = 0
se posicionó bien y ejecuta ls
                            ((pos('A',argumentos[1])<>0) or (pos('A',argumen-
tos[2])<>0) or (pos('A',argumentos[3])<>0)),    //verifica si alguno de los argu-
mentos contiene el parametro 'A'
                            ((pos('F',argumentos[1])<>0) or (pos('F',argumen-
tos[2])<>0) or (pos('F',argumentos[3])<>0)),    //verifica si alguno de los argu-
mentos contiene el parametro 'F'
```

```
                              ((pos('L',argumentos[1])<>0) or (pos('L',argumen-
tos[2])<>0) or (pos('L',argumentos[3])<>0)))    //verifica si alguno de los argu-
mentos contiene el parametro 'L'
                  else
                        begin
                        mostrarerror(4);  exit;
                        end;                                    //si <> 0 el directo-
rio recibido no es coherente.
                  end;
      end;
      analizarCD(dirBase);
End;


procedure analizarPWD(var ENTRADA: string);
{
   Nombre: analizarPWD.

}
var tipo: byte;
Begin
      if strOutPut = '' then tipo := 0 else tipo := 1;      // Tipo 0: Salida es-
tándar, tipo 1: Redirigir la salida.
      if (ENTRADA[0] >= #4) and not (flagInPut) then
            begin
            mostrarerror(2);
            exit;
            end
      else pwd(tipo);
End;


procedure analizarReDIR (var ENTRADA: string);
var   comando: string;
      i,j: word;
      ok: boolean;
      salida,respaldo:text;
      salidaRuta:string;
Begin
      i:= pos(' > ',ENTRADA);
      if i <> 0 then
            begin // > rewrite (reescribe el archivo, lo crea si no existe).
                  salidaRuta:= copy(ENTRADA,i+3,length(ENTRADA));
                  comando:= copy(ENTRADA,1,i-1);
                  crearArchivo(salida,salidaRuta,ok);
                  if not(ok) then
                        begin
                        mostrarerror(11);
                        exit;
                        end;
                  respaldarSalidaEstandar(respaldo);
                  redirigirSalidaEstandar(salida);
                  analizar(comando);
                  close(salida);
                  restaurarStdOut(respaldo);
            end
      else
      begin
            j:= pos(' >> ', ENTRADA);
```

```
            if j <> 0 then
                  begin // >> append (añade los datos al final del archivo).
                        salidaRuta:= copy(ENTRADA,j+4,length(ENTRADA));
                        comando:= copy(ENTRADA,1,j-1);
                        agregarArchivo(salida,salidaRuta,ok);
                        if not ok then
                              begin
                              Mostrarerror(12);
                              exit;
                              end;
                        respaldarSalidaEstandar(respaldo);
                        redirigirSalidaEstandar(salida);
                        analizar(comando);
                        restaurarStdOut(respaldo);
                  end;
      end;
End;


procedure analizarPipe(var ENTRADA: string);
var i: word;
      preString, postString: string;
      respaldo,commandPipe:text;
      ok:boolean;
Begin
      i:= pos(' | ',ENTRADA);
      preString   := copy(ENTRADA, 1, i-1);
      postString  := copy(ENTRADA, i+3, length(ENTRADA));
      crearArchivo(commandPipe,'commandPipe',ok);
            if not(ok) then
                  begin
                  mostrarerror(13);
                  exit;
                  end;
      popen (commandPipe,postString,'W');
      respaldarSalidaEstandar(respaldo);
      redirigirSalidaEstandar(commandPipe);
      analizar(preString);
      restaurarStdOut(respaldo);
      Pclose(commandPipe);
      deletefile('commandPipe');
End;

procedure analizarBG(var ENTRADA: string);
var i:word;
Begin
      alBG:= true;
      i:=pos(' &',ENTRADA);
      ENTRADA:= copy(ENTRADA,1,i-1);
      analizar(ENTRADA);
end;
END.


UNIT comandos;
INTERFACE

Uses BaseUnix, Unix, utilidades, users, ALR;
```

```
procedure cat     (var dir1,dir2: string; tipo: byte);
            // CAT - Concatena hasta dos archivos, o un archivo con la salida
estandar
procedure exec    (param1: String; param2: Array of AnsiString; tipo:byte);  //
EXEC - Ejecuta un programa externo. Ruta relativa o absoluta.
procedure kill    (signal, proc: longint);
                  // KILL - Envía una señal a un proceso.
procedure ls (modoA,modoF,modoL:boolean);
procedure lsAF (modoA,modoF:boolean);
                  // LS - Lista los archivos de un determinado directorio.
procedure lsL(modoA,modoF,modoL:boolean);
                  // variante de LS, lista archivos en formato largo.
procedure pwd     (tipo: byte);
                        // PWD - Muestra el directorio actual de trabajo.
procedure bg(ENTRADA:string);
procedure fg (ENTRADA:string);
procedure help;
                              //Muestra una pantalla de ayuda
procedure moo(entrada:string);

IMPLEMENTATION


//Comando CAT
procedure cat(var dir1,dir2: string; tipo: byte);
var f1,f2: text;
      texto: string;
Begin

      {$I-}       // Evita generar código de control de entrada/salida en el
programa
      assign(f1,dir1);
      reset(f1);
      if IOResult <> 0 then
            begin
            Mostrarerror(10);
            exit;
            end
      else
            begin
            while not eof(f1) do
            begin
                  readln(f1,texto);
                  if tipo = 0 then writeln(texto) else writeln(stdOutPut,texto);
            end;
            close(f1);
            end;
      if dir2 <> '' then
      begin
            assign(f2,dir2);
            reset(f2);
            if IOResult <> 0 then
                  begin
                  Mostrarerror(10);
                  exit;
                  end
            else
                  begin
                  while not eof(f2) do
```

```
                        begin
                              readln(f2,texto);
                              if tipo = 0 then writeln(texto) else
writeln(stdOutPut,texto);
                        end;
                        close(f2);
                  end;
      end;

      {$I+}       // Habilita la generación de código de entrada/salida
End;


//Comando CD
{
      El comando CD es ejecutado directamente
      por el analizadorCD en la UNIT analizador

}


//Comando EXEC
procedure exec (param1: String; param2: Array of AnsiString; tipo:byte);
var pidP,op: longint; proceso:t_procesos;

Begin
      op:= 0;
      pidP:= fpfork;
            case pidP of
            -1: begin
                  Mostrarerror(3);
                  exit;
                  end;
            0:    Begin
                        fpExecLE(param1,param2,envp);
                  end;
            else
                  begin
                        with proceso do
                              begin
                              numero:=TablaJobs.Indice;
                              nombre:=nombreComandoDesdeRuta(param1);
                              pid:=pidP;
                              estado:='Corriendo';
                              prioridad:=' ';
                              directorio:=param1;
                              end;
                        if alBG then
                              begin
                              proceso.estado:=proceso.estado+' &';
                              insertarEnTabla(proceso);
                              fpWaitPid(pidP,op,WNOHANG); //No "Espera" por el
hijo con pid=PidP
                              end
                        else
                              begin
                              pidenejec:=pidP;        // Cargo el pid del nuevo
programa como en ejecucion actual
```

```
                                fpWaitPid(pidP,op,0); //Espera por el hijo con
pid=PidP
                                if not procesoFinalizado(op) then
                                      insertarEnTabla(proceso);
                                pidenejec:=-1;          //borro el pid asignado an-
tes
                                end;
                        end;
                end;
End;


//Comando KILL
procedure kill (signal, proc: longint);
Begin
      fpKill(proc,signal);
End;


//Comando LS


procedure ls (modoA,modoF,modoL:boolean);
{
      Segun los parametros booleanos distingue entre dos tipos de LS
      y dentro de los mismos actua acorde a dichas variables
}
begin
if modoL then
      lsL(modoA,modoF,modoL)
else
      lsAF(modoA,modoF);
end;


procedure lsAF(modoA,modoF:boolean);
{
      Los archivos marcados como inaccesibles son aquellos
      en los cuales fpStat devuelve un error. En dicho caso se puede saber
      el nombre del archivo, pero no datos referidos al tipo de archivo.

}

var   directorio: pdir;
      entrada: PDirent;
      vector: vDirent;
      indice,j: integer;
      K: byte;
      info: stat;
      auxNombre,tipo    : string;

Begin
      k:= 1;tipo:='';
      indice:= 0;
      directorio := fpOpenDir(dirActual);
      if directorio <> nil  then
      begin //openDir
      repeat
            entrada:= fpReadDir(directorio^);
```

```
            if (entrada <> nil) and (modoA or (entrada^.d_name[0] <> '.')) then
//Si  mostrarTodos (ModoA) o no Oculto, lo carga al vector de directorio
                  begin
                  inc(indice);
                  vector[indice]:= entrada^;
                  end;
      until entrada = nil;
      if modoA then // si ModoA ordena el directorio alfabeticamente
            burbujaDirent(vector,indice);
      for J:= 1 to indice do
            Begin //for
            if fpStat(pchar(vector[J].d_name),info)=0 then
                  Begin  //fpStat
                  if modoA then
                        tipo:=tipoArchivoAF(info.st_mode)+' ';
                  auxNombre:=tipo+copy(vector[J].d_name,1,20);
                  case K of //case
                  1:    begin
                              if j <> indice then
                                    write(auxNombre+espacio(24-length(auxNom-
bre)))
                              else
                                    writeln(auxNombre)
                        end;
                  2:    begin
                              if j <> indice then
                                    write(auxNombre+espacio(24-length(auxNom-
bre)))
                              else
                                    writeln(auxNombre)
                        end;
                  3:    begin
                              writeln(auxNombre+espacio(24-length(auxNombre)))
                        end;
                  end; // case
                  if k = 3 then k:= 1 else inc(K);
            End // fpStat
            else
            Begin // else fpStat
                  if modoA then
                        tipo:=tipoArchivoAF(info.st_mode)+' ';
                  auxNombre:=tipo+copy(vector[J].d_name,1,20);
                  case K of //case
                  1:    begin
                              if j <> indice then
                                    write('Inaccesible: '+auxNombre+espacio(24-
length(auxNombre)))
                              else
                                    writeln('Inaccesible: '+auxNombre);
                        end;
                  2:    begin
                              if j <> indice then
                                    write('Inaccesible: '+auxNombre+espacio(24-
length(auxNombre)))
                              else
                                    writeln('Inaccesible: '+auxNombre)
                        end;
                  3:    begin
```

```
                              writeln('Inaccesible: '+auxNombre+espacio(24-len-
gth(auxNombre)))
                        end;
                  end; //case
                  if k = 3 then k:= 1 else inc(K);
            end; // else fpStat
      End; // for
      fpCloseDir(directorio^);
      end  // OpenDir
      else
            begin
            mostrarerror(4);  exit;
            end;
End;



//comando LS -l
procedure lsL(modoA,modoF,modoL:boolean);
{
      En caso de falla obteniendo los datos de un archivo se mostrará
      el mensaje "No se pudo mostrar.", que representa una falla en fpStat.

}

var   directorio  : Pdir;
      entrada           : PDirent;
      info        : stat;
      vector            : vDirent;
      contArchivos,loop: integer;
      indice,j    : integer;
      tamString,unidad  : string;
      tamanio: int64;

Begin
      indice:= 0;
      contArchivos:= 0;       //número de archivos listados
      directorio:= fpOpenDir(dirActual);
      if directorio <> nil then
      begin //openDir
            repeat
                  entrada:= fpReadDir(directorio^);
                  if (entrada <> nil) and ((modoA) or (entrada^.d_name[0] <>
'.')) then//Si  mostrarTodos (ModoA) o no Oculto, lo carga al vector de directo-
rio
                  Begin
                        inc(indice);
                        vector[indice]:= entrada^;
                  end;
            until entrada = nil;
            if modoA then // si ModoA ordena el directorio alfabeticamente
                  burbujaDirent(vector, indice);
            for J:= 1 to indice do
                  begin //for
                  if fpStat(pchar(vector[J].d_name),info)=0 then
                        Begin //fpStat
                              contArchivos:= contArchivos + 1;
                              tamanio:=info.st_size;
                              write(modoACadena(info.st_mode));
```

```
                                    write(permisosACadena(pchar(vector[J].d_name)));
                                    write(tamAsString(info.st_nlink,2),espacio(1));
                                    write(getusername(info.st_uid)+espacio(2));
                                    write(getgroupname(info.st_gid)+espacio(1));

                                    loop:=1;
                                    if (tamanio div 1024) > 1000 then
                                          begin
                                          repeat
                                                if (tamanio div 1024) > 1000 then
                                                      begin
                                                      tamanio:=tamanio div 1024;
                                                      end;
                                                inc(loop);
                                          until tamanio < 100000;
                                          end;
                                    case loop of
                                          1:unidad:=' bytes';
                                          2:unidad:='Kbytes';
                                          3:unidad:='Mbytes';
                                          4:unidad:='Gbytes';
                                          end;

                                    tamString:=' '+tamAsString(tamanio,6)+unidad;
                                    write(tamString+espacio(15-length(tamString)));
                                    write(tipoArchivoL(info.st_mode)+espacio(2));
                                    write(tiempoUnixAHumano(info.st_mtime)+espa-
cio(2));
                                          writeln(pchar(vector[J].d_name));
                              End // fpStat
                              else
                                    writeln('No se pudo mostrar.');
                        end; //for
                        writeln('- - - - - - - - - - - - - - - - - -');
                  if contArchivos = 0 then
                        writeln('No hay listado.')
                  else
                        writeln('Nro. de archivos listados: ',contArchivos);
                  fpCloseDir(directorio^);
            end //openDir
            else begin
                  mostrarerror(4);  exit;
                  end;
End;



//Comando PWD
procedure pwd(tipo: byte);
var pid, op: longint;
Begin
      pid:=fpFork;
      op:= 0;
      case pid of
      -1:begin
            mostrarerror(3);  exit;
            end;
       0:   begin
                  writeln(dirActual);
```

```
                fpKill(fpGetPid,9);
            end;
        else
            begin
                fpWaitPid(pid,op,op); //Espera por cualquier proceso hijo.
            end;
        end;
End;

procedure bg (ENTRADA:string);
var
        numeroTrabajo: longint;
        pid,error:longint;
begin
        val(ENTRADA,numeroTrabajo, error);
        pid:=damePid(numeroTrabajo);
        if pid = -1 then
            writeln('Proceso no Encontrado')
        else
            begin
                fpkill(pid,SIGCONT);
            end;
end;

procedure fg (ENTRADA:string);
var
        numeroTrabajo: longint;
        pid,error,pid2:longint;
        estado:longint;
begin
        val(ENTRADA,numeroTrabajo, error);  //Entrada es SIEMPRE un numero si
quiere funcionar (numero en String)
        pid:=damePid(numeroTrabajo);  //Obtengo el pid correspondiente al numero
de trabajo
        if pid = -1 then // No se encontro numero de trabajo
            begin
            mostrarError(20);
            exit;
            end
        else
            begin
            pidEnEjec:=pid; //cargo el programa que se va a traer al FG
            pid2:=fpfork;     // necesito que la terminal "duerma" mientras hay
otra cosa en FG
            case pid2 of
                0: begin
                        FpSetsid;   { Creo una nueva sesion para el proceso que
debe ser traido al frente
                                                http://www.freepascal.org/docs-
html/rtl/baseunix/fpsetsid.html
                                                http://stackoverflow.com/ques-
tions/9306100/how-can-i-tell-if-a-child-is-asking-for-stdin-how-do-i-tell-it-to-
stop-that}
                        fpkill(pid,SIGCONT); //Envio senial de resumen al pro-
ceso a FG
                        end;
                else
                        begin
```

```
                        fpwaitpid(pid,estado,0); //Espero que el hijo(proceso
actualmente en FG) se detenga
                        fpkill(SIGKILL,pid2);
                        ActualizarEstado(pid,estado); //actualizo el estado del
proceso que habia enviado a FG
                        pidenejec:=-1; // borro el id del proceso que se estaba
ejecutando
                        end;
                end;
        end;
end;

procedure help;
begin
      writeln('ALROShell');
      writeln();
      writeln('Copyright 2015');
      writeln('* Ramiro Rivera                    <ramarivera@gmail.com>,');
      writeln('* David lazbal                  <davidlazbal@gmail.com>,');
      writeln('* Matín Arrúa                   <martin94.profugo@gmail.com>.');
      writeln('Copyright 2014');
      writeln('* Fernando Gómez Albornoz       <fgalbornoz07@gmail.com>. ');
      writeln();
      writeln('Estas órdenes del shell están definidas internamente');
      writeln();
      writeln('bg id_trabajo');
      writeln('cat primer_archivo [segundo_archivo]');
      writeln('cd ruta');
      writeln('exit');
      writeln('fg id_trabajo');
      writeln('jobs');
      writeln('kill -id_señal id_proceso');
      writeln('ls {[-(l|f|a)]} (3 Parametros máximo) [ruta]');
      writeln('pwd');
      writeln();
      writeln('Este shell acepta los siguientes operadores: ">" ">>" "&" "|"');
      writeln();
      writeln('Para más información diríjase a la página man');
end;

procedure moo(entrada:string);
var
      aux:string;
      I:longint;
begin
(*_____________________________________
( I love Fedora, Debian? Not so much! )
 -------------------------------------
        o   ^__^
         o  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     || *)

      if upcase(ENTRADA)<>'-H' then
            begin
            if sinEspacios(Entrada)=''  then
                  aux:='Have you mooed today?'
            else
```

```
                aux:=entrada;
            write(' ');
            for I:=1 to length(aux)+2 do
                write('_');
            write(CR+LF);
            writeln('( '+aux+' )');
            write(' ');
            for I:=1 to length(aux)+2 do
                write('-');
            writeln(' ');
            writeln('         o   ^__^');
            writeln('          o  (oo)\_______');
            writeln('             (__)\       )\/\');
            writeln('                 ||----w |');
            writeln('                 ||     ||');
            writeln();
            end
      else
            begin
            writeln(space(30)+'Historia de Moo y los Super Cow Powers');
            writeln('Very well internet, you win. Let me tell you a tale about
cow powers. Super ones to be specific.');
            writeln('Once a long time ago a developer was known for announcing
his presence on IRC with a simple, to the point '+chr(39)+'Moo'+chr(39)+'.');
            writeln('As with cows in pasture others would often Moo back in
greeting. This led to a certain range of cow based jokes.');
            writeln('When apt-get was initially developed I put the enigmatic
tag line in the help message, but I did not add the '+chr(39)+'apt-get
moo'+chr(39)+' command.');
            writeln('That act lies with another, who decided that the help
teaser needed some extra zip. Thus the easter egg was born.');
            writeln('The items in aptitude are probably a homage, as aptitude
was substantially based on apt'+chr(39)+'s library.');
            writeln('It seems very popular, it was featured in Linux Magazine
some time ago, and I'+chr(39)+'ve even had people request a Moo when they find
me at conferences.');
            writeln('There have been bug reports to remove it, explain it, and
to improve the cow.');
            writeln('It was mentioned for a while  in Wikipedia, and now appar-
ently on stack exchange.');
            writeln('Now, if you look closely, in the right places, you can find
other software with cow powers. Good luck :)');
            writeln(space(50)+'-Jason Gunthorpe');
            writeln();
            end;
end;

END.


UNIT utilidades;
INTERFACE
      USES  strings, baseunix, sysutils, dateutils;

      Const       N = 300;                // Número máximo de archivos a listar
en un directorio (ls)
                  P=3;                    // Número máximo de parámetros para
los comandos (sin contar directorios)
                  LF = #10;               // Caracter Line Feed.
```

```
                CR = #13;               // Caracter Carriage Return.
                CRLF = CR + LF;         // CRLF

      Type
            vDirent = array [1..N] of Dirent;   // Vector para almacenar nombres
de archivos de un directorio y así ordenarlos.
            parametros= array [1..P] of AnsiString;        // Vector para alma-
cenar parámetros de comandos.

      var   stdOutPut: Text;              // Archivo de redirección de la salida
estándar.
            strOutPut: string;                  // Nombre del archivo de redirec-
ción.
            newInPut : Text;              // Redirección de la entrada estándar
para comando pipe.
            flagInPut: Boolean;                 // Se le asigna momentaneamente
true durante un pipe.


//  UTILIDADES
      function puntos(s: string): integer;
                  // Cantidad de puntos '.' en un string.
      function dirActual: string;
                  // Devuelve la dirección de trabajo actual.
      function dirHome: string;
                        // Devuelve el directorio Home. Por defecto /home/user.
      function strSin(param1, param2: string): string;
            // elimina param2 de la cadena param1.
      procedure burbujaDirent(var v: vDirent; max: integer);
            // ordena los archivos listados por ls. Algoritmo burbuja.
      function redireccion(s: string): boolean;
            // Devuelve true si encuentra los operadores '>' o '>>' en la ca-
dena.
      function tuberia(s: string): boolean;
                  // Devuelve true si encuentra el operador '|' en una cadena.
      function enBg(s: string): boolean;
                  // Devuelve true si el proceso debe ejecutarse en background.
      function solicitudBG(var s: string):boolean;
            // Verifica si se ingreso "bg", la solicitud para ejecutar un tra-
bajo en segundo plano.
      function solicitudFG(var s: string):boolean;
            // Verifica si se ingreso "fg", la solicitud para ejecutar un tra-
bajo en primer plano.
      procedure Mostrarerror(n: integer);
                  // Devuelve el mensaje de error <n>.
      function asString(V:int64):string;
                  // Convierte un valor a string.
      function tiempoUnixAHumano (tUnix:int64):string;
            // Convierte el valor de tiempo que devuelve Unix.
      function permisosACadena(path:string):string;
            // Verifica si el usuario tiene permisos de lectura, escritura y
ejecución, y lo imprime.
      function tamAsString(tam:int64; espacios: byte):string;
            // Permite alinear un string.
      function extraerComando(var s:string):string;
            // Devuelve el comando de la cadena (o primer componente antes del
primer espacio), y se elimina el componente de la entrada recibida.
```

```
      function extraerArgumentos(var s:string;var i:integer):parametros;
      // Devuelve un vector de argumentos de la forma -* y se eliminan los argu-
mentos de la entrada recibida.
      function modoACadena(modo:integer):string;
                  // Traduce los permisos de un archivo
      function tipoArchivoAF(modo:integer):string;
            // Traduce los tipos de archivos para el comando LS con parámetros A
y/o F
      function tipoArchivoL(modo:integer):string;
                  // Traduce los tipos de archivos para el comando LS con pará-
metro L
      function nombreComandoDesdeRuta(s:string):string;
            // Dada una dirección absoluta a un comando, devuelve sólo el nombre
del mismo.
      function paramValido(str:string):boolean;
            // Determina si un parámetro del ls es válido o no.
      function stringToAnsiString(str:string):ansiString;
            // Convierte un string en un ansiString
      function ansiStringToString(str: ansiString):string;
      // Convierte un ansiString en un string
      function sinEspacios(s:string):string;
                  // Elimina los espacios existentes en una cadena
      function eliminarSubcadena(cadena,subcadena:string):string;
      // Elimina una subcadena de la cadena original

IMPLEMENTATION

      function eliminarSubcadena(cadena,subcadena:string):string;
      var auxStr:string;
            posi:longint;
      begin
            auxStr:=cadena;
            posi:=0;
            posi:=pos(upcase(subcadena),upcase(cadena));
            if posi<>0 then
                  delete(auxStr,posi,length(subcadena));
            eliminarSubcadena:=auxstr;
      end;

      function sinEspacios(s:string):string;
      var strAux:string; I:longint;
      begin
            straux:='';
            for I:=1 to length(s) do
                  begin
                  if s[I]<>#32 then
                        strAux:=strAux+s[I];
                  end;
            sinEspacios:=strAux;
      end;

      function asString(V:int64):string;
      var S:string;
      begin
            STR(V,S);
            asString:=S;
      end;
```

```
	function puntos(s: string): integer;
	var I: integer;
	Begin
		puntos:= 0;
		for I:= 1 to length(s) do
			if s[I] = #46 then puntos:= puntos + 1;
	End;


	function dirActual: string;
	Begin
		getDir(0,dirActual);
	End;


	function dirHome: string;
	Begin
		dirHome:= fpGetEnv('HOME');
	End;


	function strSin(param1, param2: string): string;
	Begin
		strSin:= rightStr(param1,((length(param1))-(length(param2))));
	End;


	procedure burbujaDirent(var v: vDirent; max: integer);
	var I,J: integer;
		aux: Dirent;
	Begin
		for I:= 1 to max-1 do
		 for J:= 1 to max-I do
		  if upcase(strPas(v[J].d_name)) > upcase(strPas(v[J+1].d_name))
then
			Begin
				aux := v[J+1];
				v[J+1]:= v[J];
				v[J] := aux;
			End;
	End;


	function redireccion(s: string): boolean;
	Begin
		redireccion:= (pos(' > ',s) <> 0) or (pos(' >> ',s) <> 0);
	End;


	function tuberia(s: string): boolean;
	Begin
		tuberia:= (pos(' | ',s) <> 0);
	End;

	function enBg(s: string): boolean;
	Begin
		enBg:= ((length(s) <> 1) and (pos(' &',s) = length(s)-1)); // Debe
ser lo ultimo escrito
	End;
```

```
	function solicitudBG(var s: string):boolean;
	Begin
		solicitudBG:= 'BG'= upcase(s);
	End;

	function solicitudFG(var s: string):boolean;
	begin
		solicitudFG:='FG'=upcase(s);
	end;

	procedure Mostrarerror(n: integer);
	var
		errorAux:string;
	Begin
		case n of
		1:		errorAux:= 'Error n° 1: Parámetros insuficientes.';
		2:		errorAux:= 'Error n° 2: No se puede reconocer el co-
mando.';
		3:		errorAux:= 'Error n° 3: Falla en el Fork, no es posible
ejecutar el nuevo proceso.';
		4:		errorAux:= 'Error n° 4: El directorio no existe, o no
puede ser accedido.';
		5:		errorAux:= 'Error n° 5: Comando no reconocido.';
		6:		errorAux:= 'Error n° 6: Error en Exec, no es posible
ejecutar el proceso.';
		7:		errorAux:= 'Error n° 7: El programa no se encuentra en
el Path.';
		8:		errorAux:= 'Error n° 8: Parámetros incorrectos.';
		9:		errorAux:= 'Error n° 9: Los parámetros deben ser de tipo
numérico.';
		10:	errorAux:= 'Error n° 10: Error al intentar abrir el/los ar-
chivo(s).';
		11:	errorAux:= 'Error n° 11: Error de redirección de la salida es-
tándar - Operador ''>''';
		12:	errorAux:= 'Error n° 12: Error de redirección de la salida es-
tándar - Operador ''>>''';
		13:	errorAux:= 'Error n° 13: Error al intentar crear tuberías.';
		20:		errorAux:= 'Error n° 20: Trabajo no encontrado.';
		else	errorAux:= 'Error desconocido.';
		end;
		writeln(errorAux);
		if n in [1,2,5,8] then
			writeln('Si necesita ayuda utilice el comando "help".');
		exit;
	End;

function numAMes(numMes:word):string;
var
	Aux:string;
begin
	case numMes of
		1:	Aux:='ene';
		2:	Aux:='feb';
		3:	Aux:='mar';
		4:	Aux:='abr';
		5:	Aux:='may';
		6:	Aux:='jun';
		7:	Aux:='jul';
```

```
            8:    Aux:='ago';
            9:    Aux:='sep';
            10:   Aux:='oct';
            11:   Aux:='nov';
            12:   Aux:='dic';
            end;
numAMes:=aux;
end;

function tiempoUnixAHumano (tUnix:int64):string;
var
      Y,Mo,D,H,Mi,S,MS : Word;
      Aux:string;
begin
      DecodeDateTime(UnixToDateTime(tUnix),Y,Mo,D,H,Mi,S,MS);
      Aux:=numAMes(Mo)+#32;
      if D<10 then
            Aux:=Aux+#32+asString(D)+#32
      else
            Aux:=Aux+asString(D)+#32;
      if Y<2014 then
            Aux:=Aux+#32+asString(Y)+#32
      else
            begin
            if H<10 then
                  Aux:=Aux+'0'+asString(H)+':'
            else
                  Aux:=Aux+asString(H)+':';
            if Mi<10 then
                  Aux:=Aux+'0'+asString(Mi)+#32
            else
                  Aux:=Aux+asString(Mi)+#32;
            end;
      tiempoUnixAHumano:=Aux;
end;

function permisosACadena(path:string):string;
var
      Aux:String;
begin
      Aux:='---';
      if fpAccess (path,R_OK)=0 then Aux[1]:='r';
      if fpAccess (path,W_OK)=0 then Aux[2]:='w';
      if fpAccess (path,X_OK)=0 then Aux[3]:='x';
      permisosACadena:=Aux+#32;
end;

function tamAsString(tam:int64; espacios: byte):string;
var
      aux,buff:string;
      i:integer;
begin
      buff:='';
      aux:=asString(tam);
      for i:=1 to espacios-length(aux) do
            buff:=buff+' ';
      tamAsString:=buff+Aux+#32;
end;
```

```
function extraerComando(var s:string):string;
var aux:string;e:integer;
begin
      while (s<>'') and (s[1]=' ') do
            delete(s,1,1);                              //eliminar espacios ini-
ciales
      if (s<>'') then
      begin
            e:=1;
            aux:='';
            repeat                                            //comienza detección
de primer componente
            begin
                  aux:=aux + s[e];                            //agrega un caracter
                  if s[e] = '"' then                                //si el ultimo
elemento agregado es ".
                  begin
                        inc(e);
                        while s[e] <> '"' do                  //agrega todo hasta
el proximo ".
                        begin
                              aux:=aux + s[e];
                              inc(e);
                        end;
                        aux:=aux + s[e];                      //agrega las " de
cierre
                  end;
                  inc(e);
            end;
            until (e>length(s))OR(s[e] = ' ');                          //se en-
contró un espacio fuera de ""
            extraerComando:= aux;                       //devolver comando
            s:=copy(s,length(aux)+1,length(s)); //la entrada se recorta hasta el
primer parametro
            while (s<>'') and (s[1]=' ') do
                  delete(s,1,1);                              //eliminar espacios
iniciales
      end
      else
            extraerComando:=s;
end;

function extraerArgumentos(var s:string;var i:integer):parametros;
var j:integer;param:ansiString;
begin
      for j:=1 to P do
            extraerArgumentos[j]:='';           //inicializar vector de argumen-
tos
      while (s<>'')AND(s[1]=' ') do
            delete(s,1,1);                              //eliminar espacios inicia-
les
      i:=0;
      param:='';
      while (i<=N)AND(s<>'')AND(s[1]='-') DO          //comienza detección de
primer componente
      begin
            delete(s,1,1);          //elimina el guion que indica parametro
            WHILE (s<>'')AND(s[1]<>' ') DO
            begin
```

```
                param:=param + s[1];
                delete(s,1,1);
            end;
            inc(i);
            extraerArgumentos[i]:=param;
            param:='';
            WHILE (s<>'')AND(s[1]=' ') DO
                delete(s,1,1);                          //eliminar espacios
entre parametros
      end;
end;

function modoACadena(modo:integer):string;
var
      Aux:String;
begin
      Aux:='----------';
      if (modo and S_IFDIR) = S_IFDIR then Aux[1]:='d';
      if (modo and S_IRUSR) = S_IRUSR then Aux[2]:='r';
      if (modo and S_IWUSR) = S_IWUSR then Aux[3]:='w';
      if (modo and S_IXUSR) = S_IXUSR then Aux[4]:='x';
      if (modo and S_IRGRP) = S_IRGRP then Aux[5]:='r';
      if (modo and S_IWGRP) = S_IWGRP then Aux[6]:='w';
      if (modo and S_IXGRP) = S_IXGRP then Aux[7]:='x';
      if (modo and S_IROTH) = S_IROTH then Aux[8]:='r';
      if (modo and S_IWOTH) = S_IWOTH then Aux[9]:='w';
      if (modo and S_IXOTH) = S_IXOTH then Aux[10]:='x';
      modoACadena:=Aux+#32;
end;

function tipoArchivoAF(modo:integer):string;
begin
      if fpS_ISLNK(modo) then
            tipoArchivoAF:=('(L)')
            else
                  if fpS_ISCHR(modo) then
                        tipoArchivoAF:=('(C)')
                  else
                        if fpS_ISREG(modo) then
                              tipoArchivoAF:=('(F)')
                        else
                              if fpS_ISDIR(modo) then
                                    tipoArchivoAF:=('(D)')
                                    else
                                          if fpS_ISBLK(modo) then
                                                tipoArchivoAF:=('(B)')
                                          else
                                                if fpS_ISFIFO(modo) then
                                                      tipoArchivoAF:=('(I)')
                                                else
                                                      if fpS_ISSOCK(modo) then
                                                            tipoAr-
chivoAF:=('(S)');
end;


function tipoArchivoL(modo:integer):string;
begin
      if fpS_ISLNK(modo) then
```

```
            tipoArchivoL:=('LNK'+#32)
            else
                  if fpS_ISCHR(modo) then
                        tipoArchivoL:=('CHR'+#32)
                  else
                        if fpS_ISREG(modo) then
                              tipoArchivoL:=('FILE')
                        else
                              if fpS_ISDIR(modo) then
                                    tipoArchivoL:=('DIR'+#32)
                              else
                                    if fpS_ISBLK(modo) then
                                          tipoArchivoL:=('BLK'+#32)
                                    else
                                          if fpS_ISFIFO(modo) then
                                                tipoArchivoL:=('FIFO')
                                          else
                                                if fpS_ISSOCK(modo) then
                                                      tipoArchivoL:=('SOCK');
end;

function nombreComandoDesdeRuta(s:string):string;
var aux:string;
begin
      aux:= s;
      while (pos('/',aux) <> 0) do
            aux:=copy(aux,pos('/',aux)+1,length(aux));
      nombreComandoDesdeRuta:=aux;
end;

function paramValido(str:string):boolean;
begin
      str:=UPCASE(str);
      paramValido:=(
                              (str='A') or (str='F') or (str='L') or
                              (str='LA') or (str='LF') or (str='AL') or
(str='AF') or (str='FL') or (str='FA') or
                              (str='LFA') or (str='LAF') or (str='FLA') or
(str='FAL') or (str='AFL') or (str='ALF')
                        )
end;

function stringToAnsiString(str:string):ansiString;
begin
      stringToAnsiString:=str;
end;

function ansiStringToString(str: ansiString):string;
begin
      ansiStringToString:=str;
end;

BEGIN
      strOutPut:= '';                           // StrOutPut refiere a la redi-
rección de la salida estándar.
      setTextLineEnding(stdOut,CRLF); // Asignándole formato a la salida están-
dar.
      flagInPut:= false;
```

```
END.


unit ALR;

interface

uses baseUnix, Unix, Linux, errors,sysutils,utilidades;


type puntero=^nodo;

       t_procesos=record
            nombre:string;
            prioridad:string;
            numero:longint;
            directorio:string;
            estado:string;
            pid:longint;
            end;

       nodo=record
                  info:t_procesos;
                  sig:puntero;
                  end;

       tabla=record
                  cab:puntero;
                  tam:word;
                  indice: word;
                  end;
var

            alBG: boolean;                      // El programa debe ejecutarse en
bg.
            tablaJobs: Tabla;             // Tabla de tareas activas.
            pidEnEjec: longint;                 // PID del proceso en primer
plano.


procedure agregarArchivo(var arch: text; filename:string; var ok:boolean);
procedure abrirArchivo(var arch: text; filename:string; var ok:boolean);
procedure crearArchivo(var arch: text; filename:string;var ok:boolean);
procedure redirigirSalidaEstandar(var arch:text);
procedure respaldarSalidaEstandar(var respaldo:text);
procedure restaurarStdOut(var arch:text);

(*-------------------------------------------------------------------------
-----*)


function AnalizarEstado(pid:longint;estado:longint):string;
function procesoFinalizado(estado:longint):boolean;
function codigoFinalizacionProceso(estado:longint):longint;
function procesoSenializado(estado:longint):boolean;
function senialRecibidaPorProceso(estado:longint):longint;
procedure ActualizarEstado(pid:longint;estado:longint);
Procedure SIGTSTP_Recibida(sig : cint);cdecl;
Procedure SIGINT_Recibida(sig : cint);cdecl;
```

```
procedure SIGCHLD_Recibida(signal: LongInt; info: psiginfo; context: PSigCon-
text);
procedure instalarManejadores;

(*-------------------------------------------------------------------------
-----*)

procedure crearTabla;
procedure insertarEnTabla (x:t_procesos);
procedure eliminarDeTabla (pid: longint ;var eliminado:t_procesos);
procedure mostrarTabla;
function damePid (numero:longint):longint;
procedure eliminarPorEstado (var ok:boolean;var correcto:boolean; var elimi-
nado:t_procesos);
procedure limpiarTabla;
function espacio (n:byte):string;
function procesoEnBlanco():t_procesos;
function encontrarProceso (numeroProceso: longint):t_procesos;
function MostrarUno(X:T_procesos):string;


implementation


Procedure SIGTSTP_Recibida(sig : cint);cdecl;
begin
      if pidenejec<>-1 then
            fpkill(SIGTSTP,pidenejec)
end;


Procedure SIGINT_Recibida(sig : cint);cdecl;
begin
      if pidenejec<>-1 then
            fpkill(SIGINT,pidenejec)
end;


procedure SIGCHLD_Recibida(signal: LongInt; info: psiginfo; context: PSigCon-
text);
{
      Procedimiento a realizar al recibir una sekal SIGCHLD ,
      la cual se recibe cuando el estado de un proceso hijo es modificado.
      Este procedimiento analiza la causa de la modificacion y la refleja en la
Tabla de JOBS
}
begin
      ActualizarEstado
            (
            info^._sifields._sigchld._pid,      //Pid del hijo que envio la se-
nial
            info^._sifields._sigchld._status    // Estado de terminacion para ser
analizado
            );

            {
                  Estructura de psiginfo
                        type psiginfo = ^tsiginfo;
```

```
                Estructura de tsiginfo (en nuestro caso longint = 4)

                      type tsiginfo =
                      record
                            si_signo: LongInt;            Signal number
                            si_errno: LongInt;            Error code
                            si_code: LongInt;       Extra code (?)
                            _sifields: record       Extra signal information
fields
                                  case LongInt of
                                  0:    (
                                         _pad: array [0..(SI_PAD_SIZE)-1] of
LongInt;          Padding element
                                        );
                                  1:    (
                                        _kill:      record
      Signal number (or status)
                                                    _pid: pid_t;
      Sending process ID
                                                    _uid: uid_t;
      Sending User ID
                                                    end;
                                        );
                                  2:    (
                                        _timer: record                      De-
fault timer
                                                    _timer1: DWord;
      Timer 1 (system time)
                                                    _timer2: DWord;
      Timer 2 (user time)
                                                    end;
                                        );
                                  3:    (
                                        _rt:  record
      Posix compatibility record
                                                    _pid: pid_t;
            Sending process ID
                                                    _uid: uid_t;
            Sending User ID
                                                    _sigval: pointer;
      Signal value
                                                    end;
                                        );
                                  4:    (
                                        _sigchld:   record
      SIGCHLD signal record
                                                          _pid: pid_t;
            Sending process ID
                                                          _uid: uid_t;
            Sending User ID
                                                          _status: LongInt;
      Signal number (or status, SIGCHLD)
                                                          _utime: clock_t;
      User time
                                                          _stime: clock_t;
      System time
                                                          end;
                                        );
                                  5:    (
```

```
                                        _sigfault:  record
      SIGILL, SIGFPE, SIGSEGV, SIGBUS record
                                                    _addr: pointer;
            Address (SIGILL, SIGFPE, SIGSEGV, SIGBUS)
                                                    end;
                                        );
                                  6:    (
                                        _sigpoll:   record
      SIGPOLL record
                                                    _band: LongInt;
            SIGPOLL band
                                                    _fd: LongInt;
            SIGPOLL file descriptor
                                                    end;
                                        );
                                  end;
                        end;
            }
end;


procedure instalarManejadores;
{
      Instala manejadores de señales personalizados para las señales
      SIGCHLD ,SIGTSTP y SIGINT
}
var oldAction,newAction : PSigActionRec; //Punteros hacia registros de acciones
ante seniales
begin
      new(newAction);
      new(oldAction);
      newAction^.sa_Handler:=SigActionHandler(@SIGCHLD_Recibida); //Asigno el
handler a la nueva senial como la funcion SIGCHLD_Recibida
      fillchar(newAction^.Sa_Mask,sizeof(newAction^.sa_mask),#0); //Inicializa-
cion
      newAction^.Sa_Flags:=SA_SIGINFO; {o 4}    //La senial debera devolver in-
formacion detallada, otras acciones son IGNORAR, o DEFAULT
      {$ifdef Linux}              //Compatiblidad Linux
            newAction^.Sa_Restorer:=Nil;
      {$endif}
      if (fpSigAction(SIGCHLD,newAction,oldAction)<>0)then //Chequeo de errores
durante instalacion del nuevo manejador en newAction, el antiguo manejador se
guarda en oldAction
            begin
            writeln('Error en Instalacion: ',fpgeterrno,'.');
            halt(1);
            end;
      fpSignal(SIGTSTP,SignalHandler(@SIGTSTP_Recibida)); //Instalo manejador de
SIGTSTP
      fpSignal(SIGINT,SignalHandler(@SIGINT_Recibida));     //Instalo manejador
de SIGINT
      {
            fpSignal tiene un subconjutno de la funcionalidad de fpSigAction -->
menos funciones
            No necesito tanta informacion sobre el contexto de la recepcion de
TSTP o INT, solo que accion realizar
      }
end;
```

```
function procesoFinalizado(estado:longint):boolean;
begin
      procesoFinalizado:= (wifexited(estado)) ;
end;

function procesoSenializado(estado:longint):boolean;
begin
      procesoSenializado:= (wifsignaled(estado)) ;
end;

function senialRecibidaPorProceso(estado:longint):longint;
begin
      senialRecibidaPorProceso:=wtermsig(estado);
end;

function codigoFinalizacionProceso(estado:longint):longint;
begin
      codigoFinalizacionProceso:=(WExitstatus(estado));
end;


function AnalizarEstado(pid:longint;estado:longint):string;
{
      Analiza el estado devuelto por un fpwaitpid, en funcion
      de la razon de terminacion y/o seniales recibidas por el child
}
begin
if procesoFinalizado(estado) then
                  begin //El programa finalizo por si mismo
                  AnalizarEstado:='Finalizado('+asString(codigoFinalizacionPro-
ceso(estado))+')'; //obtengo su codigo de terminacion (puede haber terminado en
error)
                  end
            else
                  begin
                  if (procesoSenializado(estado)) then
                        begin //El programa envio SIGCHLD por cambio de estado
                        case senialRecibidaPorProceso(estado) of //analizo la
senial que produjo el cambio

                        SIGKILL,
                        SIGTERM,
                        SIGINT:     begin
                                    AnalizarEstado:='Terminado';
                                    fpkill(SIGKILL,pid);
                                    end;

                        SIGCONT:begin
                                    AnalizarEstado:='Corriendo';
                                    end;

                        SIGSTOP,
                        SIGTSTP,
                        SIGTTIN,
                        SIGTTOU:begin
                                    AnalizarEstado:='Detenido';
```

```
                                    end;

                              end;
                              end;
                        end;
end;


procedure ActualizarEstado(pid:longint;estado:longint);
var
      Aux:T_procesos;
begin
      eliminarDeTabla(pid,Aux);
      Aux.Estado:=AnalizarEstado(pid,estado);
      insertarEnTabla(Aux);
      dec(tablaJobs.Indice);
end;



(*-------------------------------------------------------------------------
-----*)

procedure agregarArchivo(var arch: text; filename:string; var ok:boolean);
begin
      {$I-}
      assign(arch,filename);
      append(arch);
      ok:=(IOResult=0);
      if not ok then
            crearArchivo(arch,filename,ok);
      {$I+}
end;

procedure abrirArchivo(var arch: text; filename:string; var ok:boolean);
begin
      {$I-}
      assign(arch,filename);
      reset(arch);
      ok:=(IOResult=0)
      {$I+}
end;

procedure crearArchivo(var arch: text; filename:string;var ok:boolean);
begin
      {$I-}
      assign(arch,filename);
      rewrite(arch);
      ok:=(IOResult=0)
      {$I+}
end;

procedure redirigirSalidaEstandar(var arch:text);
{
      el FD Output (Abstraccion de FPC para el FD 1 = STDOUT)
      apunta ahora al ARCH.
}
begin
      fpdup2(arch,output)
```

```
end;

procedure respaldarSalidaEstandar(var respaldo:text);
{
Duplico y devuelvo el FD de Output para luego restaurarlo
}
var
      ok:boolean;
begin
      crearArchivo(respaldo,'salidaEstandarRespaldo',ok);
      if not ok then
            begin
            writeln('Error Critico');
            halt;
            end;
      fpdup2(output,respaldo);
end;

procedure restaurarStdOut(var arch:text);
{
Restaura la STDOUT al FD original, el cual DEBE HABER SIDO PREVIAMENTE RESPAL-
DADO
}
begin
      close(Output);
      assign(Output,'');
      rewrite(output);
      redirigirSalidaEstandar(arch);
      deletefile('salidaEstandarRespaldo');
end;


(*-------------------------------------------------------------------------
-----*)

function asString(V:longint):string;
{Convierte un número en una cadena}
var S:string;
begin
      STR(V,S);
      asString:=S;
end;

procedure crearTabla;
{Crea e inicializa la tablaJobs}
begin
      tablaJobs.cab:=nil;
      tablaJobs.tam:=0;
      tablaJobs.indice:=0;
end;

procedure insertarEnTabla (x:t_procesos);
{Agregar un elemento proceso a la tablaJobs, ordenando los mismos por número}
var dir,ant,act: puntero;
begin
      new(dir);
      dir^.info:=x;
      if (tablaJobs.cab=nil) or (tablaJobs.cab^.info.numero>x.numero) then
            begin
```

```
            dir^.sig:=tablaJobs.cab;
            tablaJobs.cab:=dir;
            end
      else
            begin
            ant:=tablaJobs.cab;
            act:=tablaJobs.cab^.sig;
            while (act<>nil) and (act^.info.numero < x.numero) do
                  begin
                  ant:=act;
                  act:=act^.sig;
                  end;
            ant^.sig:=dir;
            dir^.sig:=act;
            end;
      inc(tablaJobs.tam);
      inc(tablaJobs.indice);
end;

procedure eliminarDeTabla (pid: longint;var eliminado:t_procesos);
{Elimina un elemento proceso de la tablaJobs, según un pid ingresado}
var ant,act: puntero;
begin
      if tablaJobs.cab<>nil then
      begin
      if tablaJobs.cab^.info.pid = pid then
            begin
                  act:=tablaJobs.cab;
                  tablaJobs.cab:=tablaJobs.cab^.sig;
                  eliminado:=act^.info;
                  dispose(act);
                  dec(tablaJobs.tam);
            end
      else
            begin
            ant:=tablaJobs.cab;
            act:=tablaJobs.cab^.sig;
            while (act<>nil) and (act^.info.pid <> pid) do
                  begin
                  ant:=act;
                  act:=act^.sig;
                  end;
            if act <> nil then
                  begin
                  ant^.sig:=act^.sig;
                  eliminado:=act^.info;
                  dispose(act);
                  dec(tablaJobs.tam);
                  end;
      end;
end;
end;


procedure eliminarPorEstado (var ok:boolean;var correcto:boolean; var elimi-
nado:t_procesos);
{Elimina un elemento proceso de la tablaJobs si su estado actual es "Finalizado"
o "Terminado"}
var ant,act: puntero;
```

```
begin
      ok:=false;
      correcto:=false;
      if tablaJobs.cab<>nil then
      begin
            if (pos('Finalizado',tablaJobs.cab^.info.estado)<>0) or (pos('Termi-
nado',tablaJobs.cab^.info.estado)<>0) then
                  begin
                        eliminado:=tablaJobs.cab^.info;
                        act:=tablaJobs.cab;
                        tablaJobs.cab:=tablaJobs.cab^.sig;
                        dispose(act);
                        dec(tablaJobs.tam);
                        correcto:=true;
                  end
            else
                  begin
                  ant:=tablaJobs.cab;
                  act:=tablaJobs.cab^.sig;
                  while (act<>nil) and ((pos('Finalizado',act^.info.estado)=0)
and (pos('Terminado',act^.info.estado)=0)) do
                        begin
                        ant:=act;
                        act:=act^.sig;
                        end;
                  if act <> nil then
                        begin
                        ant^.sig:=act^.sig;
                        eliminado:=act^.info;
                        dispose(act);
                        dec(tablaJobs.tam);
                        correcto:=true;
                        end
                  else
                        ok:=true;
                  end;
            end
end;


function MostrarUno(X:T_procesos):string;
{Muestra la información de un elemento proceso de la tablaJobs}
begin

MostrarUno:='['+(asString(x.numero))+']'+x.prioridad+espacio(7-len-
gth(asString(x.numero)))+x.estado+espacio(16-length(x.estado))+x.nombre+espa-
cio(5)+'('+(asString(x.pid))+')';

end;


procedure mostrarTabla;
{Muestra todos los elementos proceso de la tablaJobs}
var aux:puntero;
begin
      if tablaJobs.cab <> nil then
      begin
            writeln('JobID',espacio (5),'Estado',espacio (10),'Nombre');
            aux:=tablaJobs.cab;
```

```
            repeat
                  writeln(MostrarUno(aux^.info));
                  aux:=aux^.sig;
            until aux=nil;
      end
      else
            writeln('No hay trabajos.');
end;

function damePid (numero:longint):longint;
{Devuelve el pid de un elemento proceso según un número ingresado}
var act:puntero;
begin
      act:=tablaJobs.cab;
      while (act<>nil) and (act^.info.numero<>numero) do
      begin
            act:=act^.sig;
      end;
      if act <> nil then
            damePid:=act^.info.pid
      else
            damePid:=-1;
end;


procedure limpiarTabla;
{Eliminado todos los elementos proceso cuyo estado sea "Finalizado" o "Termi-
nado"}
var ok,correcto: boolean; eliminado: t_procesos;
begin
      ok:=false;
      correcto:=false;
      while not(ok) and (tablaJobs.cab <> nil) do
            begin
            eliminarPorEstado(ok,correcto,eliminado);
            if ((correcto) and (eliminado.pid <> 0)) then
                  writeln(MostrarUno(eliminado));
            end;
end;


function espacio ( n: byte):string;
{Inserta tantos espacios como indique "n"}
var i:byte;
      str:string;
begin
      str:='';
      for i:=1 to n do
            str:=str+' ';
      espacio :=str;
end;


function encontrarProceso(numeroProceso: longint):t_procesos;
{Busca un elemento proceso según un número ingresado}
var aux: puntero; encontrado: boolean; x:t_procesos;
begin
      encontrarProceso:=procesoEnBlanco;
      if tablaJobs.cab <> nil then
```

```
      begin
            encontrado:=false;
            aux:=tablaJobs.cab;
            repeat
                  x:=aux^.info;
                  if x.numero = numeroProceso then
                        begin
                        encontrado:=true;
                        encontrarProceso:=x;
                        end;
                  aux:=aux^.sig;
            until (encontrado) or (aux=nil);
      end;
end;

function procesoEnBlanco():t_procesos;
{"Inicializa" los campos de un elemento proceso}
var x:t_procesos;
begin
      with x do
      begin
            numero:=0;
            nombre:='<< Null process >>';
            pid:=-1;
            estado:='';
            prioridad:='';
            directorio:='';
      end;
      procesoEnBlanco:=x;
end;


end.
```

Conclusión.

La implementación de las funciones previamente citadas puede no haber sido realizada en su totalidad teniendo en cuenta "todas" las "reglas y normas" previamente instruidas en nosotros, ya que (en este caso particular, Free Pascal, siendo un lenguaje perteneciente al paradigma imperativo estructurado) en más de una ocasión las necesidades funcionales requerían un quiebre respecto a la utopía de un programa escrito acorde a un "Paradigma de Programación Puro". A esto debemos sumarle las limitaciones y prestaciones propias del lenguaje y del sistema, las cuales, para nosotros como desarrolladores, se traducían en la cantidad, flexibilidad y potencia de las llamadas al sistema disponibles en FPC (Un ejemplo de esto, es el sacrificio de la UNIT CRT, la cual brinda control avanzado de Sonido, Pantalla y Teclado; a cambio de "Cierta peRdida de conTrol").
Luego del estudio de la catedra, los distintos trabajos prácticos y la realización de este proyecto, tenemos un mayor panorama del funcionamiento de los sistemas operativos con los que "convivimos" todos los días, y de la difícil tarea que implica su construcción y correcto funcionamiento. Lo anterior es, debido en gran parte a la orientación de los trabajos prácticos y las condiciones impuestas sobre el trabajo practico final, especialmente cierto para sistemas Unix, y siendo aún más específicos aquellos cuyas distros se hallan basadas en Debian (Debian, Ubuntu, Lubuntu, Linux Mint, etc.).
Fue satisfactoriamente interesante pasar del enfoque teórico (el cual en algunos momentos se tornaba tedioso debido a la dificultad para el alumno promedio de "conectar" la teoría dictada por los profesores con la verdadera implementación y funcionamiento de un S.O.) a un enfoque más practico que nos permitió realmente aplicar gran parte de lo aprendido durante la asignatura. Probablemente antes teníamos una mera noción del esfuerzo que conlleva implementar (Y hacerlo bien, aun mas importante) un Sistema Operativo; la cual ahora podemos asegurar ha ganado profundidad.